Bullet Points:

- C# code is case sensitive
- Razor comment: @* your comment *@
- All new razor pages must be added inside the *Pages* folder
- *_Layout.cshtml* is a partial view which encapsulates all razor pages inside the *Pages* folder
- Handler methods must be public. They can return either void or IActionResult
- An html form must have a method attribute, and it must be equal to post.

```
<form method="post">
```

- Add an **asp-page-handler** attribute to the form if there are multiple OnPost() methods in one page
- To inject the DB class -> at program.cs file add

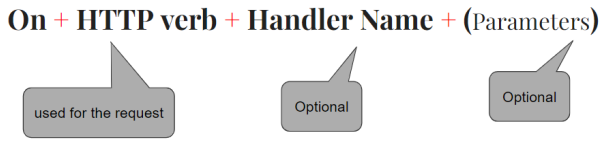
1. Add namespace:
   using solutionName.Models;
2. **builder.Services.AddSingleton<DB>();**

Tag Helpers:

1. asp-route-id
2. asp-page-handler
3. asp-for
4. asp-page
5. asp-validation-for

Structure of handler methods:

- Handler methods must use the following naming convention:




```
public void OnGet(){}

public void OnPost(){}
```

ADO.net

Connecting web app with database using Ado.net:

1. Include necessary packages
   ```
   using System.Data;
   using System.Data.SqlClient;
   ```
2. open a connection SqlConnection
   ```
   string constr = “”
   SqlConnection con = new
   SqlConnection(constr);
   con.Open();
   ```
3. Form a query
   ```
   string query = “”
   ```
4. Form a SqlCommand (cmd)
   ```
   SqlCommand cmd = new SqlCommand(query,
   con);
   ```
5. Execute the command
   a. cmd.ExecuteReader() -> (Returns table)
   b. cmd.ExecuteNonQuery() -> (Returns 1 row affected)
   c. cmd.ExecuteScalar() -> (Returns single value)
6. Catch any exceptions (optional)
7. Close the connection
   ```
   con.Close();
   ```

DataTable

- needed namespace:
  Using System.Data
- Property:
  public DataTable dt { get; set; }
- Object:
  DataTable dt = new DataTable();

Take Input From the User

(To pass parameters from the frontend to the backend)

Frontend:

```
<input type="text"  asp-for="@Model.Name">
<input type="text"  asp-for="@Model.Email">
```

Backend:

```
[BindProperty]
public string Name { get; set; }
[BindProperty]
public string Email { get; set; }
```

Sending parameters between pages

1. at OnPost():
   return RedirectToPage("/Index", new {x = Name});
   where: x -> on the index page & name -> in the current page
2. at index.cs:
   [BindProperty(SupportsGet =true)]
   public string x{ get; set; }

Model Validation:

1. Data Annotation Attributes + Properties (Error Message=””) -> Table (Entity) Class
2. If(ModelState.IsValid) -> OnPost()
3. -> .cshtml form

DataAnnotations

- [Required]
- [Range(1, 100)]
- [StringLength(60, MinimumLength = 3)]

Bootstrap Tables

```
<table class="table">
    <thead>
        <tr>
            <th scope="col">#</th>
            <th scope="col">Name</th>
            <th scope="col">Password</th>
        </tr>
    </thead>
    <tbody>
        @for (int i = 0; i < @Model.dt.Rows.Count; i++)
        {
            <tr>
                <th scope="row">@i</th>
                @for (int j=0; j<@Model.dt.Columns.Count; j++)
                {
            <td>@Model.dt.Rows[i][j]</td>
                }
            </tr>
        }
    </tbody>
</table>
```

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

Bootstrap Forms

```
<form>
  <div class="form-group">
    <label for="exampleInputEmail1">Email address</label>
    <input type="email" class="form-control" placeholder="Enter email">
    <small id="emailHelp" class="form-text text-muted">We'll never share your email with anyone else.</small>
  </div>
  <div class="form-group">
    <label for="exampleInputPassword1">Password</label>
    <input type="password" class="form-control"placeholder="Password">
  </div>
  <div class="form-group form-check">
    <input type="checkbox" class="form-check-input">
    <label class="form-check-label" for="exampleCheck1">Check me out</label>
  </div>
  <button type="submit" class="btn btn-primary">Submit</button>
</form>
```